

CODING FORMS FOR SRC INDEXING

Microfiche No.	OTS0558032		
New Doc ID	96940001770	Old Doc ID	
Date Produced	12/08/93	Date Received	08/08/94
		TSCA Section	8D
Submitting Organization	BASF CORP		
Contractor			
Document Title	DIPHENYLMETHANE, 4,4'-DIISOCYANATE - A THREE-DIMENSIONAL MODEL OF THE VAPOR CONCENTRATIONS OVER AN MDI SPILL, WITH ATTACHMENTS AND COVER LETTER DATED 1/29/94		
Chemical Category	DIPHENYLMETHANE, 4,4'-DIISOCYANATE (101-68-8)		

CODING FORM FOR GLOBAL INDEXING

REV. 7/27/82

Microfiche No. (7) *	1	No. of Pages	2
Doc I.D.	3	Old Doc I.D.	4
Case No.(s)			5
Date Produced (6)	6	Date Rec'd (6)	7
		Conf. Code *	8
Check One: <input type="checkbox"/> Publication <input type="checkbox"/> Internally Generated <input type="checkbox"/> Externally Generated			
Pub/Journal Name	9		
9			
Author(s)	10		
Organ. Name	11		
Dept/Div	12		
P.O. Box	13	Street No./Name	14
City	15	State	16
		Zip	17
		Country	18
MID No. (7)	19	D & B NO. (11)	20
Contractor	21		
Doc Type	22		
• • • • • 8.D • • • • •			
Doc Title	23		
Chemical Name (300 per name)	25	CAS No. (10)	24

BASF Corporation

BASF

Certified Mail
Return Receipt Requested

July 29, 1994

Attn: 8(d) Health and Safety Reporting Rule
Document Processing Center (TS-790)
Rm. L-100
Office of Pollution Prevention and Toxics
Environmental Protection Agency
401 M St. SW
Washington, DC 20460

Contains No CBI

94 AUG -8 AM 7:50

RECEIVED
OPPT/ATC

Dear Ladies and Gentlemen:

Subject: Health and Safety Reporting Rule
Diphenylmethane, 4,4'-diisocyanate
(CAS #101-68-8)

Enclosed is a report entitled "A Three-Dimensional Computer Model of the Vapor Concentrations Over an MDI Spill" and an abstract entitled "Model for Predicting Environmental Exposure From an MDI Spill." The abstract is intended to be presented at the Society of Plastics Polyurethane Conference to be held in Boston in October, 1994.

The report is based upon a computer modeling study developed by BASF Corporation to confirm industry's position on the appropriate personal protective equipment needed to respond to small discharges of Diphenylmethane, 4,4'-diisocyanate (4,4'-MDI) (CAS #101-68-8). Historical area and personal monitoring indicated that the American Conference of Governmental Industrial Hygienists (ACGIH) Threshold Limit Value (TLV) and the Occupational Safety and Health Administration (OSHA) Permissible Exposure Ceiling Level (PEL-C) of 5 ppb and 20 ppb respectively are not exceeded. The computer model was used to predict the location of the TLV and PEL concentrations around various spills of 4,4'-MDI. Worst-case conditions for a small spill were assumed: no air movement, no reaction with atmospheric moisture, and only molecular diffusion.

The abstract concludes that 4,4'-MDI diffusion is slow. PEL and TLV exposure levels stay close to the center of the spill and do not expand far from the edge. The model was based on a worst-case scenario for 4,4'-MDI, therefore, exposure levels to polymeric MDI should be less. The report concludes that respiratory protection is not needed.

Although BASF is uncertain whether the report and abstract constitute an 8(d) health and safety study, since the model confirms monitoring data and is based on a worst case scenario for a small spill, BASF believes that EPA will find the material interesting and useful.



86940001770

Page 2

Environmental Protection Agency
Washington, DC

Submitting site: BASF Corporation
8 Campus Drive
Parsippany, NJ 07054

Submitting Official: Carol E. Sunman
Product Regulations Specialist
BASF Corporation
1609 Biddle Avenue
Wyandotte, MI 48192-3799

No claims of confidentiality are made in this submission. Please contact me at (313) 246-5121 if there are any questions regarding the modeling study.

Sincerely,

BASF CORPORATION



Carol E. Sunman
Product Regulations Specialist

enc.

Contains No CBI

**"MODEL FOR PREDICTING ENVIRONMENTAL EXPOSURE
FROM A MDI SPILL"**

Abstract Number: 033
Session: Mb
Day: Wednesday, October 12

William P. Robert and Gregory Roginski
BASF Corporation
1609 Biddle Ave.
Wyandotte, MI 48192

94 AUG -8 AM 7:50

RECEIVED
OFFICE
10

Abstract

Product Stewards are responsible for product compliance and providing related advice throughout a product's lifecycle. A problem which has perplexed Product Stewards, Industrial Hygienists and Emergency Responders for many years has been how to respond to a small discharge of Diphenylmethane 4,4' Diisocyanate (MDI). The question is: should full SCBA (Self Contained Breathing Apparatus) be used by responders, or is respiratory protection required for this purpose? To help answer this question, a three dimensional computer model was developed to predict the vapor concentrations of MDI over a known discharge. The model assumed worst case scenario of no air movement, no reaction with atmospheric moisture and only molecular diffusion. The computer model was used to predict the location of the TLV and PEL concentrations around various spills of MDI. The results of the simulation suggest MDI diffuses slowly and remains close to the surface of the spill.

Introduction

Lifecycle analysis is an integral part of a the Product Steward's evaluation of a product's potential risks and hazards. Product related advice does not end at the front gate, but should be given to the Steward's business, sites, transporters, distributors, collers, and customers (cradle to grave). The potential risks and hazards of a product are determined from a number of sources, including physical and chemical properties, literature searches, industrial and practical experiences, industrial hygiene and environmental monitoring. Most recently, the use of computer modeling has provided an excellent tool for risk evaluation.

A computer model was developed to answer the question of how to respond to discharges of MDI. The question is, should full SCBA or airline systems be used to clean up small spills, or is respiratory protection not required for this purpose? This question has been asked by not only Product Stewards, Industrial Hygienists and Emergency Responders, but also by regulatory agencies and our customers throughout the product's lifecycle. Historical area and personal monitoring within our plants, and at our customers (with the exception of spray applications), have indicated employee exposure to MDI does not exceed the American Conference of Governmental Industrial Hygienist (ACGIH) TLV®-TWA of 5 ppb, nor the Occupational Safety and Health Administration (OSHA) PEL-C of 20 ppb ceiling limit. This is supported by the fact that MDI has a very low vapor pressure and will not readily vaporize. In fact, the saturated vapor concentration of 4,4' MDI at 25°C is 0.013 ppm.

Continuing the work

Continuing the work

TLV® TWA - This is the time-weighted average concentration for a normal 8-hour work day or 40-hour work week, to which nearly all workers may be repeatedly exposed, day after day, without adverse effect.

PEL-C - This is the concentration that should not be exceeded during any part of the working exposure.

Discussion

To further clarify response to small discharges and to confirm our industry's position on personal protective equipment required for clean up, a three dimensional computer model was developed to predict the vapor concentrations of MDI over a known quantity of spilled material. The following conservative assumptions were made for the MDI spill simulation.

- The temperatures of the spill and the air are the same and remain constant.
- The total pressure remains constant.
- Only trace quantities of MDI will be present in the air.
- No air movement.
- No chemical reactions such as MDI and water vapor.
- Diffusion of MDI in the air remains constant.
- Air is not soluble in MDI.
- Both MDI and air are single components.
- The vapor pressure of Polymeric MDI is given by 4,4' MDI and is constant.
- No aerosols are present, therefore the concentration of MDI at the surface is given by the vapor pressure of MDI
- The area of the spill does not change with time.
- The spill is circular.

Mathematical calculations are included in the Appendix. The spill size was estimated using an arbitrary liquid film thickness of 2mm and the volume of the spill.

Results

Several simulation programs were run representing a five (5) gallon, fifty (50) gallon, and two hundred and fifty (250) gallon spill for the following situations:

- (1) Locate the position of the MDI TLV® (0.005ppm) around the spill at 20°C.
- (2) Locate the position of the MDI TLV® (0.005ppm) around the spill at 38°C.
- (3) Locate the position of the MDI PEL (0.02ppm) around the spill at 38°C.

The vapor pressure of MDI at 20°C results in a concentration below the MDI PEL; therefore this concentration cannot be exceeded at any time.

The results of the simulation at 20°C indicated that the TLV® concentration will reach about 0.8 feet above the center of the spill after ten (10) hours. Above the edge of the spill, the TLV®

concentration will reach a maximum of 0.7 feet. (See Figures 1,4,7).

At 38°C, the TLV® concentration will reach about 3.8 feet above the center of the spill and about 3.5-3.7 feet above the edge of the spill. (See Figures 2,5,8,) At this same temperature, the PEL concentration reaches about 2.2 feet above the center of the spill and about 1.6-1.8 feet above the edge of the spill after 10 hours. (See Figures 3,6,9)

Conclusion

The results of these simulation runs indicate that, under the assumptions used in the model,

- The diffusion of MDI from the spill is rather slow, and
- The exposure limit concentration (TLV or PEL) stays close to the surface of the spill and does not expand far from the edge of the spill, and
- The model used the vapor pressure for pure Diphenylmethane 4,4' Diisocyanate as a worst case scenario, thus the expected exposure limit concentrations from Polymeric MDI are not expected to be near these values, but less.

It is our experience that most spills reported (i.e. Chemtrec) are contained and remediated soon after discovery. Under these conditions, it appears unlikely that respiratory protection would be required to prevent excessive employee exposure to MDI vapors emitted from the spill. We would not, therefore, recommend respiratory protection.

As previously noted, computer modeling to predict the TLV® and PEL concentrations above a spill will aid in advising our facilities, transporters and customers in the proper handling of MDI, by suggesting appropriate personal protective equipment, and of the potential for "overexposure" to employees responding to the spill. This will also aid in the risk assessment of our product as part of our ongoing commitment to the Responsible Care® Initiative.

APPENDIX AEquations Used in Computer Model

For the evaporation of the material, the equation of continuity will describe the concentrations in space and time:

$$\frac{\partial \rho_i}{\partial t} = (\nabla \cdot n_i) + r_i \quad (1)$$

where

- ρ_i = mass density of the i-th component,
- n_i = mass flux of the i-th component,
- r_i = reaction rate of the i-th component and
- t = time.

Using the previously discussed spill assumptions together with Fick's law of diffusion, the equation of continuity becomes

$$\frac{\partial x_A}{\partial t} = D_{AB} \nabla^2 x_A \quad (2)$$

- where x_A = mole fraction of component A (MDI),
- D_{AB} = Diffusivity of component A (MDI) in component B (air) and
- t = time.

In cylindrical coordinates, with no angular dependence, $\nabla^2 x_A$ is given by

$$\nabla^2 x_A = \frac{1}{r} \frac{\partial x_A}{\partial r} + \frac{\partial^2 x_A}{\partial r^2} + \frac{\partial^2 x_A}{\partial z^2} \quad (3)$$

where r is the radial direction and z is the vertical direction.

The equation of continuity then is

$$\frac{\partial x_A}{\partial t} = D_{AB} \left[\frac{1}{r} \frac{\partial x_A}{\partial r} + \frac{\partial^2 x_A}{\partial r^2} + \frac{\partial^2 x_A}{\partial z^2} \right] \quad (4)$$

The boundary conditions for this problem are

- x_A is zero at $r = \infty$ and at $z = \infty$,

- x_A is equal to the vapor pressure divided by the total pressure at $r \leq R$ and $z = 0$, where R is the radius of the spill,
- there is no flux of material into the ground ($z = 0$) for $r > R$ and
- the mole fraction must be finite at the center of the spill.

The last boundary condition requires that the first term on the right hand side of equation (4) have a finite value. At the center of the spill the radial flux must be zero or else there would be a source or sink at the centerline. Using L'Hôpital's rule

$$\lim_{r \rightarrow 0} \frac{\partial x_A / \partial r}{r} = \lim_{r \rightarrow 0} \frac{\partial^2 x_A / \partial r^2}{1} = \frac{\partial^2 x_A}{\partial r^2} \quad (5)$$

Thus at the origin the equation of continuity becomes

$$\frac{\partial x_A}{\partial t} = D_{AB} \left[2 \frac{\partial^2 x_A}{\partial r^2} + \frac{\partial^2 x_A}{\partial z^2} \right] \quad (6)$$

With the following change of variables, the equation of continuity (4) may be put into dimensionless form (8). This will simplify solving the problem.

$$\chi_A = \frac{x_A}{x_{Ao}} \quad \eta = \frac{r}{R} \quad \xi = \frac{z}{R} \quad \tau = \frac{D_{AB} t}{R^2} \quad (7a,b,c,d)$$

$$\frac{\partial \chi_A}{\partial \tau} = \frac{1}{\eta} \frac{\partial \chi_A}{\partial \eta} + \frac{\partial^2 \chi_A}{\partial \eta^2} + \frac{\partial^2 \chi_A}{\partial \xi^2} \quad (8)$$

where x_{Ao} is the mole fraction at the surface of the spill. The boundary conditions become

- $\chi_A = 0$ at $\eta = \infty$ and at $\xi = \infty$
- $\chi_A = 1$ for $\eta \leq 1$ and $\xi = 0$
- $\partial \chi_A / \partial \eta = 0$ for $\eta > 1$ and $\xi = 0$.

A relatively efficient method to solve equation (8) is the Peaceman-Rachford alternating direction implicit (ADI) method [1]. This method uses central difference approximations for each of the partial derivatives and solves the partial differential equation in two stages. The time step is split in half and for the first half step an intermediate solution is calculated using the partial derivatives in only one coordinate direction. For the second half time step, the final solution is calculated using the intermediate solution and the partial derivatives in the other coordinate direction. The algebraic equations resulting from this method are easy to solve and require a small amount of computer memory.

Since the computer can only solve a finite problem, the boundary conditions at $\eta = \infty$ and $\xi = \infty$ pose a problem. It was decided to treat the upper horizontal and right side vertical edges of the simulation grid as solid surfaces with no flux allowed through those edges. This allows the concentration at the edge to change as the interior concentration changes without imposing a zero concentration at the edge of the grid. The grid was made large enough so that during the entire simulation time the concentration at the edge of the grid was not greater than 10^{-5} of the concentration directly over the spill.

An estimate of the diffusivity of MDI in air was made using the method of Fuller, et. al. [2]. This method only requires a knowledge of the structure of the components. For MDI-air the diffusion coefficient was estimated to be $0.053 \text{ cm}^2/\text{sec}$ at 20°C .

The spill size was estimated using an arbitrary liquid film thickness of 2 mm and the volume of the spill. The radius of the resulting spill is about 6 feet for 5 gallons, 17.8 feet for 50 gallons, and 40.3 feet for 250 gallons, respectively.

References

1. Peaceman, D. W., and H. H. Rachford, Jr., "The numerical solution of parabolic and elliptic differential equations," J. Soc. Indust. Appl. Math., 3, 28 (1955).
2. Fuller, E. N., Schettler, P. D. and J. C. Giddings, Ind. Eng. Chem., 58, 18 (1966).

Biographies

William P. Robert received his Masters in Occupational and Environmental Health from Wayne State University in 1981 and became a Certified Industrial Hygienist in 1990. He has worked for BASF Corporation Urethanes Group on ecology related assignments. He is currently the Manager of Product Stewardship and Regulatory Compliance for the Urethanes Group.

Gregory T. Roginski received his Ph.D. in Chemical Engineering from Wayne State University in 1981. Since then he has worked for BASF Corporation in their Research Services Department. He is part of the Proces Development group and is involved in the scale-up of various products.

RESEARCH STATUS REPORT

RESEARCH SERVICES UNIT

A THREE-DIMENSIONAL COMPUTER MODEL OF THE VAPOR CONCENTRATIONS OVER AN MDI SPILL

AUTHOR: G. ROGINSKI

8 December, 1993

SUPERVISOR: M. CAPRARO *mc*

Summary

A three-dimensional computer model of the MDI (methylenediphenyldiisocyanate) vapor concentrations over an MDI spill has been developed. This computer model uses only molecular diffusion and does not take into account air movement over the spill. The computer model was used to predict the location of the MDI TLV (Threshold Limit Value) and the PEL (Permissible Exposure Limit) concentrations around a five gallon spill of MDI. The results of the simulation suggest that MDI diffuses slowly from the spill and remains close to the surface of the spill. For example, the results of the simulation for an MDI spill at 38°C shows that the MDI TLV (0.005 ppm) rises about four feet above the center of the spill after ten hours. At the same time at ground level the TLV concentration is at a position of about eight feet from the center of the six foot radius spill.

*Here is the title page
with the approvals.*

Greg R.

51 AUG -8 AM 7:50

RECEIVED
OPI 6/10

BASF Corporation	BASF Corporation	BASF Corporation
DISTRIBUTION:	REPORT NUMBER	RSR-151
M. Capraro, J. Louvar*, W. Robert, M. Martin, R. Chwalik,	SECURITY CLASS	B
TIC	PROJECT NUMBER	42146
*Summary Only	SUPERVISOR(S)	APPROVAL(S)
	<i>J. Louvar</i>	

RESEARCH STATUS REPORT

RESEARCH SERVICES UNIT

A THREE-DIMENSIONAL COMPUTER MODEL OF THE VAPOR CONCENTRATIONS OVER AN MDI SPILL

AUTHOR: G. ROGINSKI

SUPERVISOR: M. CAPRARO

8 December, 1993

Summary

A three-dimensional computer model of the MDI (methylenediphenyldiisocyanate) vapor concentrations over an MDI spill has been developed. This computer model uses only molecular diffusion and does not take into account air movement over the spill. The computer model was used to predict the location of the MDI TLV (Threshold Limit Value) and the PEL (Permissible Exposure Limit) concentrations around a five gallon spill of MDI. The results of the simulation suggest that MDI diffuses slowly from the spill and remains close to the surface of the spill. For example, the results of the simulation for an MDI spill at 38°C shows that the MDI TLV (0.005 ppm) rises about four feet above the center of the spill after ten hours. At the same time at ground level the TLV concentration is at a position of about eight feet from the center of the six foot radius spill.

*You will receive the
summary page with
the approval signature
later. Greg R.*

BASF Corporation

BASF Corporation

BASF Corporation

DISTRIBUTION:

M. Capraro, J. Louvar*, W. Robert, M. Martin, R. Chwalik,
TIC

*Summary Only

REPORT NUMBER RSR-151

SECURITY CLASS B

PROJECT NUMBER 4P146

SUPERVISOR(S) APPROVAL(S)

1. Introduction

A simple estimate of the vapor concentration over a chemical spill can be made using the vapor pressure of the material at the given temperature. This would be the concentration at which the air would be saturated with the material. For a small spill of a material with a low volatility in a large enclosed area this concentration would probably not be easily reached. To make an estimate of how long it would take for the material to diffuse from such a spill, a three-dimensional computer model was developed. This model was applied to a simulated spill of five gallons of MDI in an enclosed space with no air movement.

2. Discussion

For the evaporation of the material, the equation of continuity will describe the concentrations in space and time:

$$\frac{\partial \rho_i}{\partial t} = -(\nabla \cdot n_i) + r_i \quad (1)$$

where ρ_i = mass density of the i-th component,
 n_i = mass flux of the i-th component,
 r_i = reaction rate of the i-th component and
 t = time.

The following assumptions were made for the MDI spill simulation:

- The temperatures of the spill and the air are the same and remain constant
- The total pressure remains constant
- Only trace quantities of MDI will be present in the air
- No air movement
- No chemical reactions such as MDI and water vapor
- Constant diffusivity of MDI in the air
- Air is not soluble in MDI
- Both MDI and air are single components
- The vapor pressure of MDI is given by 4,4'-MDI and is constant
- The concentration of MDI at the surface is given by the vapor pressure of MDI (no aerosols are present)
- The area of the spill does not change with time
- The spill is circular

With these simplifying assumptions and using Fick's law of diffusion, the equation of continuity becomes

$$\frac{\partial x_A}{\partial t} = D_{AB} \nabla^2 x_A \quad (2)$$

where x_A = mole fraction of component A (MDI),
 D_{AB} = Diffusivity of component A (MDI) in component B (air) and

t = time.

In cylindrical coordinates, with no angular dependence, $\nabla^2 x_A$ is given by

$$\nabla^2 x_A = \frac{1}{r} \frac{\partial x_A}{\partial r} + \frac{\partial^2 x_A}{\partial r^2} + \frac{\partial^2 x_A}{\partial z^2} \quad (3)$$

where r is the radial direction and z is the vertical direction.

The equation of continuity then is

$$\frac{\partial x_A}{\partial t} = D_{AB} \left[\frac{1}{r} \frac{\partial x_A}{\partial r} + \frac{\partial^2 x_A}{\partial r^2} + \frac{\partial^2 x_A}{\partial z^2} \right] \quad (4)$$

The boundary conditions for this problem are

- x_A is zero at $r = \infty$ and at $z = \infty$,
- x_A is equal to the vapor pressure divided by the total pressure at $r \leq R$ and $z = 0$, where R is the radius of the spill,
- there is no flux of material into the ground ($z = 0$) for $r > R$ and
- the mole fraction must be finite at the center of the spill.

The last boundary condition requires that the first term on the right hand side of equation (4) have a finite value. At the center of the spill the radial flux must be zero or else there would be a source or sink at the centerline. Using L'Hôpital's rule

$$\lim_{r \rightarrow 0} \frac{\partial x_A / \partial r}{r} = \lim_{r \rightarrow 0} \frac{\partial^2 x_A / \partial r^2}{1} = \frac{\partial^2 x_A}{\partial r^2} \quad (5)$$

Thus at the origin the equation of continuity becomes

$$\frac{\partial x_A}{\partial t} = D_{AB} \left[2 \frac{\partial^2 x_A}{\partial r^2} + \frac{\partial^2 x_A}{\partial z^2} \right] \quad (6)$$

With the following change of variables, the equation of continuity (4) may be put into dimensionless form (8). This will simplify solving the problem.

$$\chi_A = \frac{x_A}{x_{Ao}} \quad \eta = \frac{r}{R} \quad \xi = \frac{z}{R} \quad \tau = \frac{D_{AB} t}{R^2} \quad (7a,b,c,d)$$

$$\frac{\partial \chi_A}{\partial \tau} = \frac{1}{\eta} \frac{\partial \chi_A}{\partial \eta} + \frac{\partial^2 \chi_A}{\partial \eta^2} + \frac{\partial^2 \chi_A}{\partial \xi^2} \quad (8)$$

where x_{Ao} is the mole fraction at the surface of the spill. The boundary conditions become

- $\chi_A = 0$ at $\eta = \infty$ and at $\xi = \infty$
- $\chi_A = 1$ for $\eta \leq 1$ and $\xi = 0$
- $\partial\chi_A / \partial\eta = 0$ for $\eta > 1$ and $\xi = 0$.

A relatively efficient method to solve equation (8) is the Peaceman-Rachford alternating direction implicit (ADI) method [1]. This method uses central difference approximations for each of the partial derivatives and solves the partial differential equation in two stages. The time step is split in half and for the first half step an intermediate solution is calculated using the partial derivatives in only one coordinate direction. For the second half time step, the final solution is calculated using the intermediate solution and the partial derivatives in the other coordinate direction. The algebraic equations resulting from this method are easy to solve and require a small amount of computer memory. A detailed derivation is given in Appendix A. The computer program is listed in Appendix B.

Since the computer can only solve a finite problem, the boundary conditions at $\eta = \infty$ and $\xi = \infty$ pose a problem. It was decided to treat the upper horizontal and right side vertical edges of the simulation grid as solid surfaces with no flux allowed through those edges. This allows the concentration at the edge to change as the interior concentration changes without imposing a zero concentration at the edge of the grid. The grid was made large enough so that during the entire simulation time the concentration at the edge of the grid was not greater than 10^{-6} of the concentration directly over the spill.

An estimate of the diffusivity of MDI in air was made using the method of Fuller, et. al. [2]. This method only requires a knowledge of the structure of the components. For MDI-air the diffusion coefficient was estimated to be $0.053 \text{ cm}^2/\text{sec}$ at 20°C . Detailed calculations are given in Appendix C.

The spill size was estimated using an arbitrary liquid film thickness of 2 mm and the volume of the spill (5 gal). The radius of the resulting spill is about 6 feet.

3. Results

The simulation program was run for the following situations:

- (1) Locate the position of the MDI TLV (0.005 ppm) around the spill at 20°C ,
- (2) Locate the position of the MDI TLV (0.005 ppm) around the spill at 38°C , and
- (3) Locate the position of the MDI PEL (0.02 ppm) around the spill at 38°C .

The vapor pressure of MDI at 20°C results in a concentration below the MDI PEL so that this concentration cannot be exceeded at any time.

The results of the simulation at 20°C indicate that the TLV concentration will reach about 0.76 feet above the center of the spill after 10 hours. Above the edge of the spill, the TLV concentration will reach only about 0.4 feet (See Figures 1 and 2).

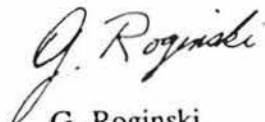
At 38°C the TLV concentration will reach about 3.8 feet above the center of the spill and about 3.3 feet above the edge of the spill. The TLV concentration extends out just beyond eight feet from the center of the spill after 10 hours (See Figures 3, 4, 5 and 6). At this same temperature, the PEL concentration reaches about 2.2 feet above the center of the spill and about 1.6 feet above the edge of the spill after 10 hours (See Figures 7 and 8).

4. Conclusions

The results of these simulation runs indicate that, under the assumptions used in the model,

- The diffusion of MDI from the spill is rather slow, and
- The exposure limit concentration (TLV or PEL) stays close to the surface of the spill and does not expand far from the edge of the spill.

This model does not include the effects of air movement over the spill. Any air currents would be expected to reduce the concentrations above the spill but increase the concentration downwind of the spill. Any air turbulence over the spill could cause the MDI to be carried higher in the air than predicted by this model.

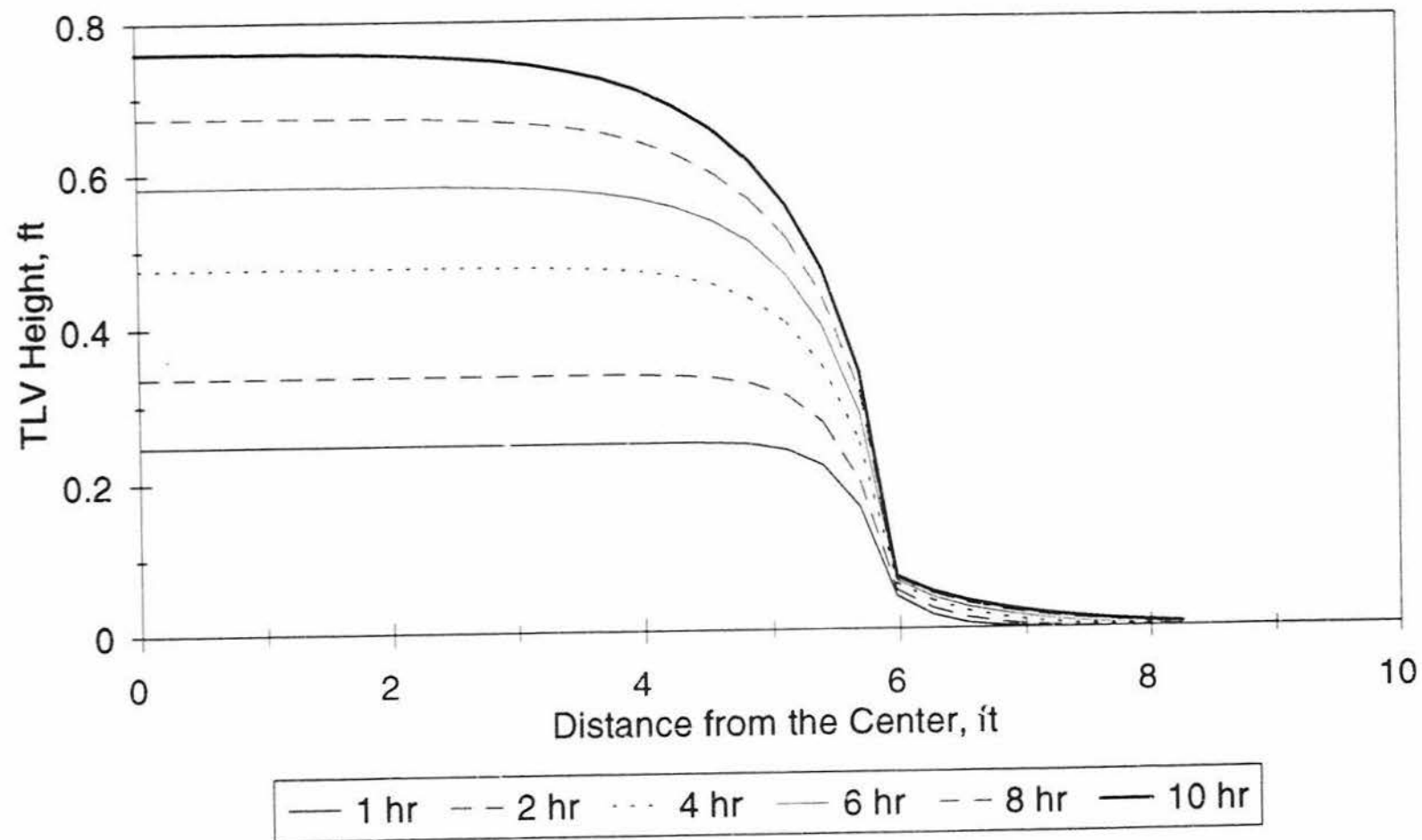

G. Roginski

References

1. Peaceman, D. W., and H. H. Rachford, Jr., "The numerical solution of parabolic and elliptic differential equations," J. Soc. Indust. Appl. Math., 3, 28 (1955).
2. Fuller, E. N., Schettler, P. D. and J. C. Giddings, Ind. Eng. Chem., 58, 18 (1966).

Position of the MDI TLV

Temperature = 20°C



Spill Radius = 5.97 ft

Figure 1
Position of the MDI TLV Concentration at 20°C - Cross Section View

Position of the MDI TLV Around an MDI Spill

Temperature = 20°C

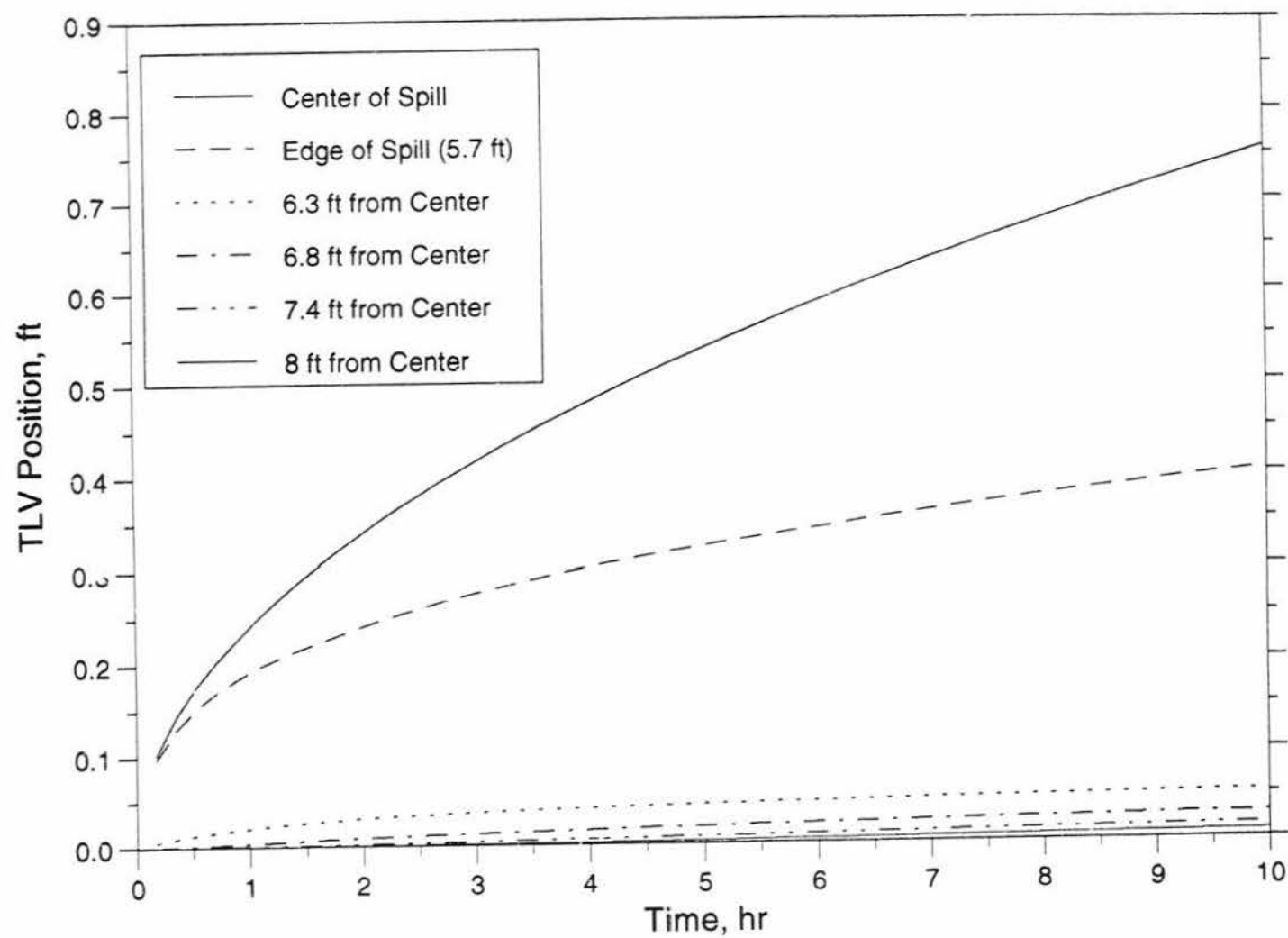
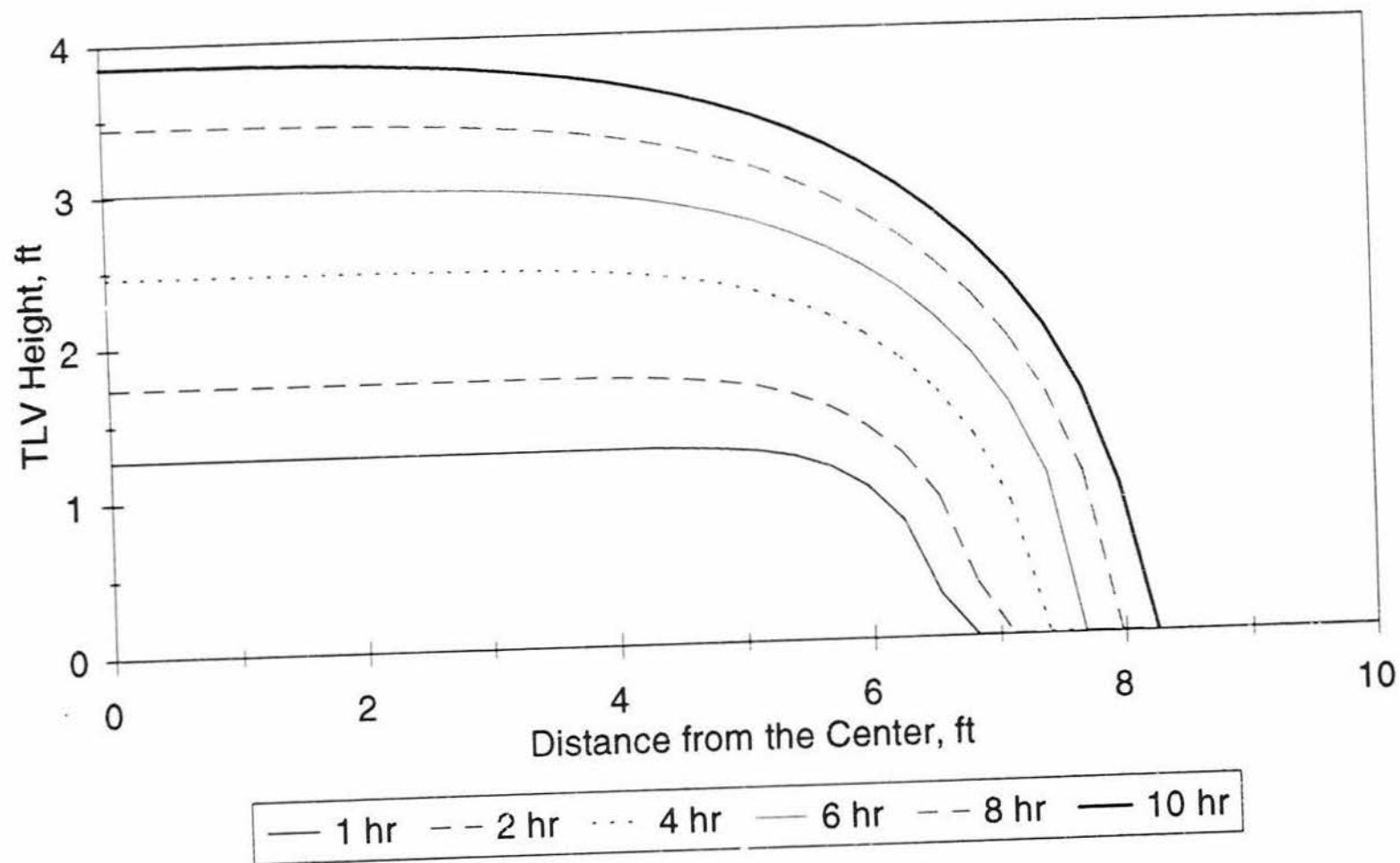


Figure 2
Position of the MDI TLV Concentration at 20°C - Temporal View

Position of the MDI TLV

Temperature = 38°C



Spill Radius = 5.97 ft

Figure 3
Position of the MDI TLV Concentration at 38°C - Cross Section View

Position of the MDI TLV Around an MDI Spill

Temperature = 38°C

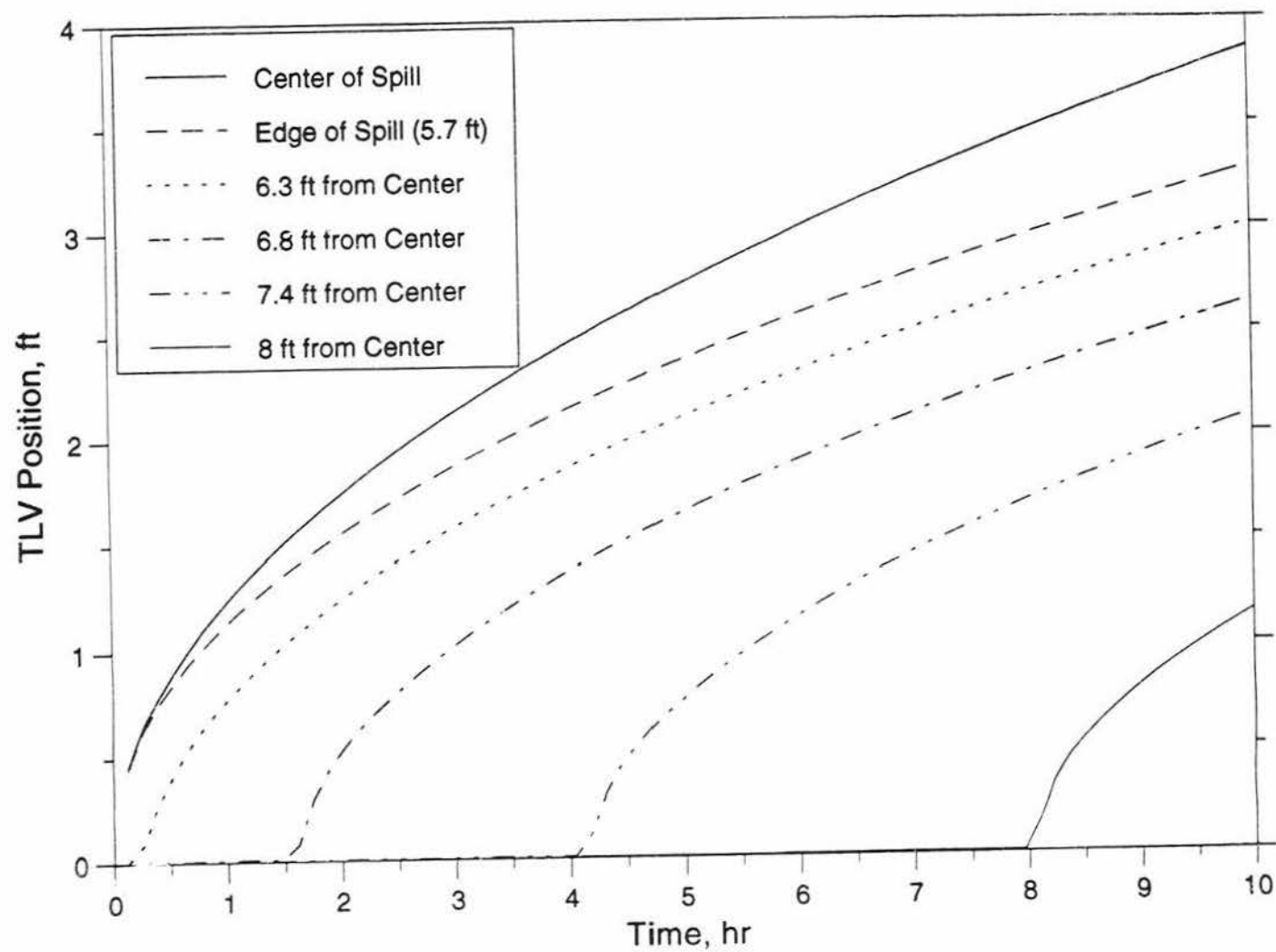


Figure 4

Position of the MDI TLV Concentration at 38°C - Temporal View

Position of the MDI TLV

Temperature = 38°C

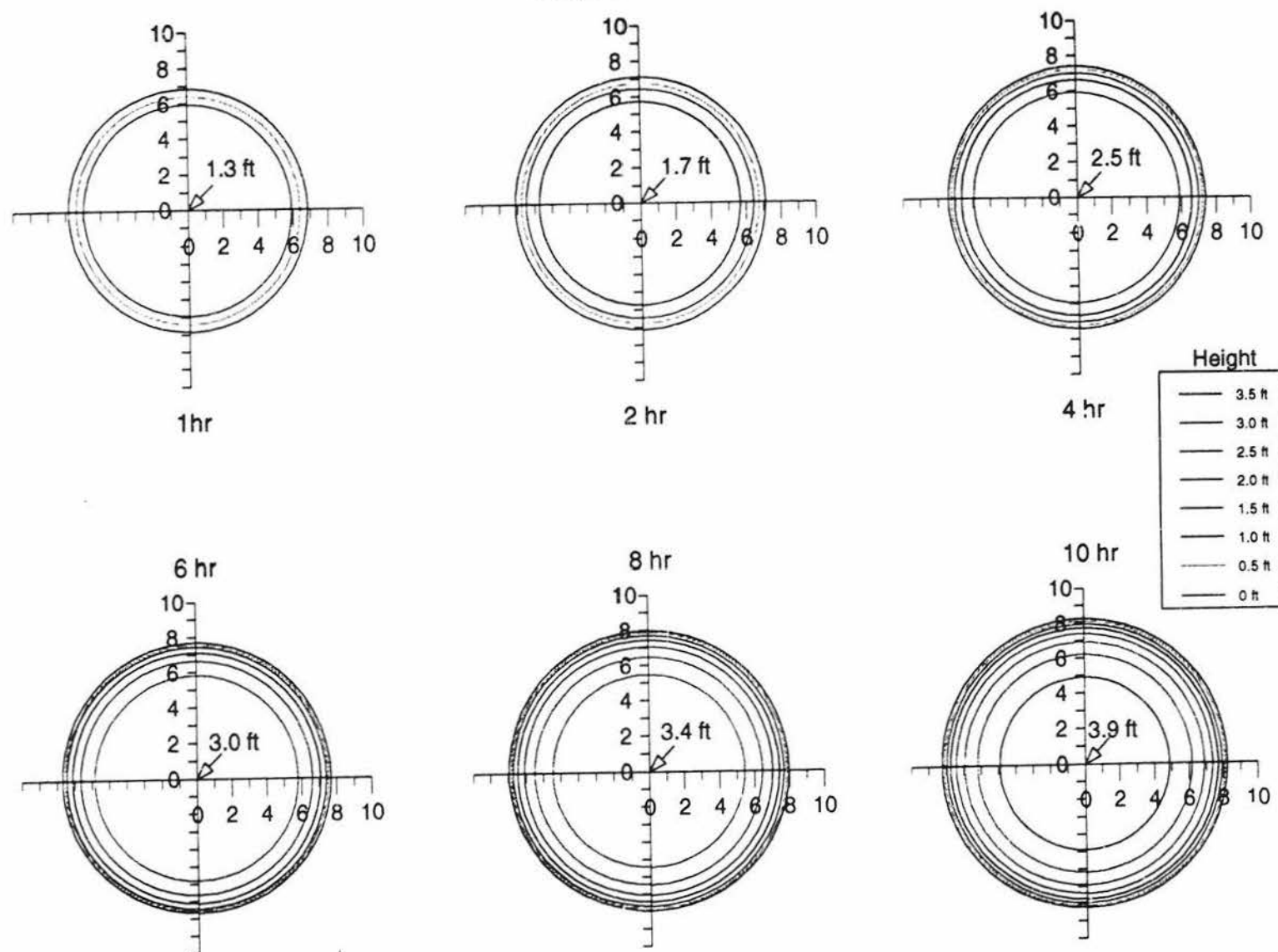


Figure 5

View of the MDI TLV Concentration at 38°C - Contour Plot

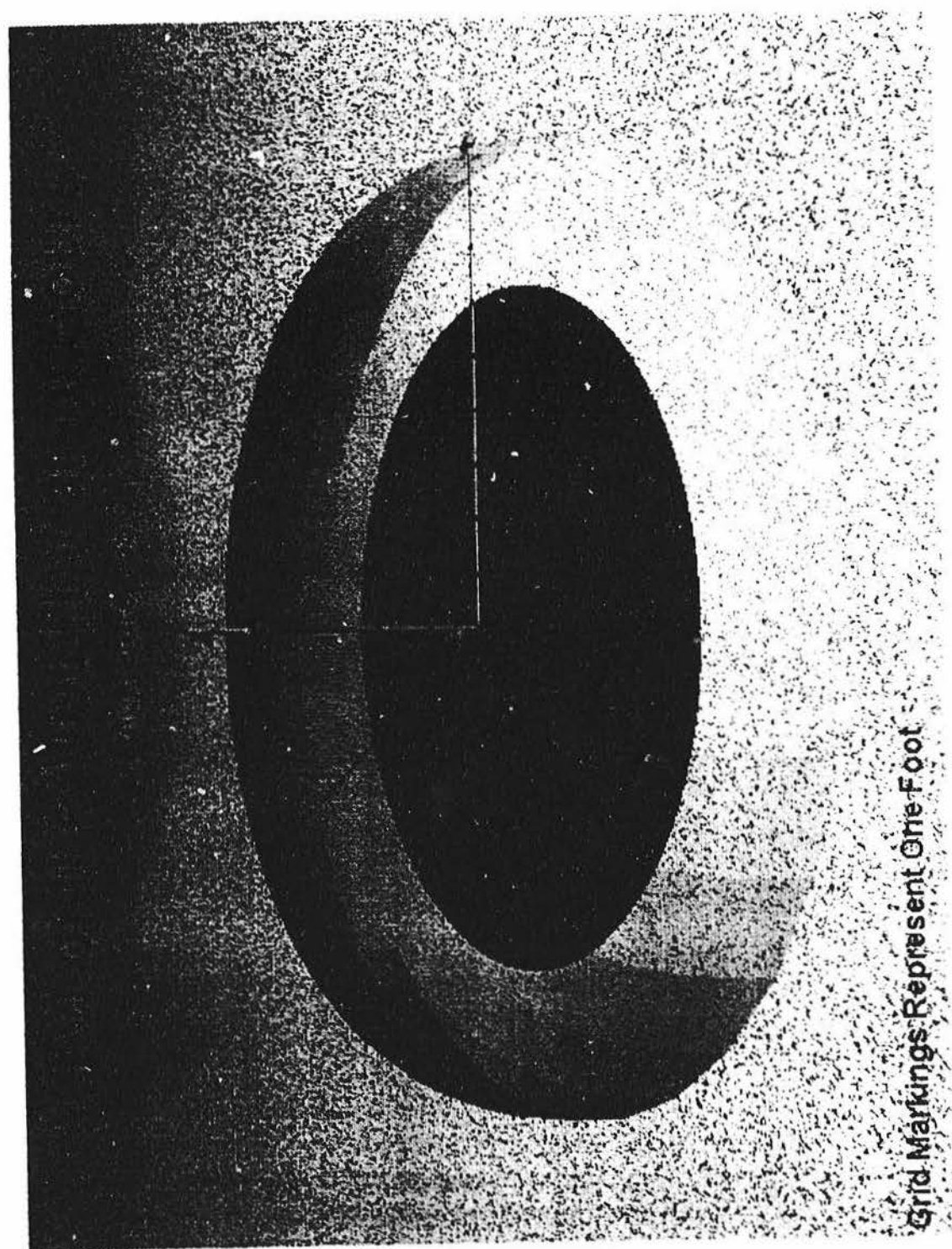


Figure 6
3-D View of the MDI TLV Concentration at 38°C

Position of the MDI PEL

Temperature = 38°C

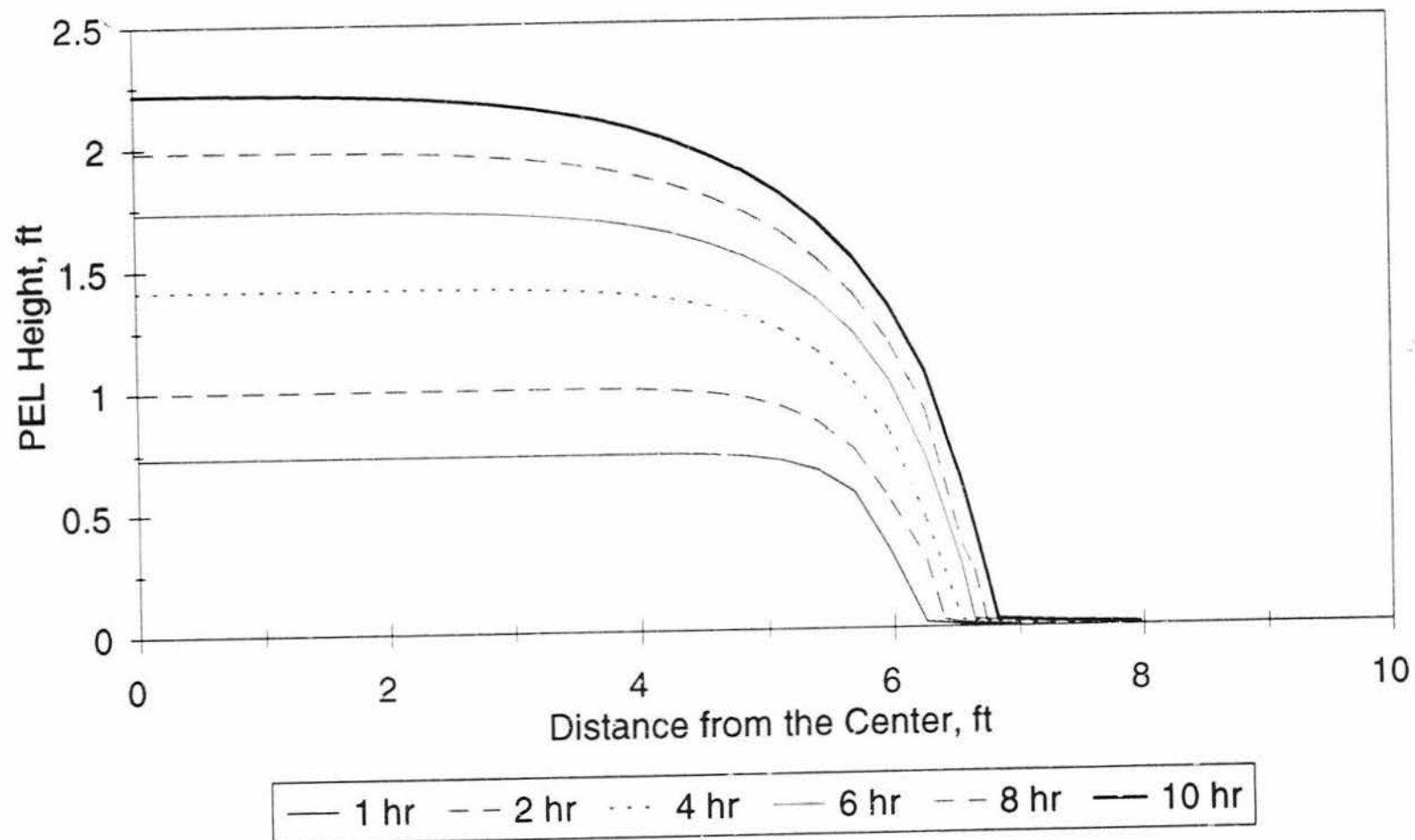


Figure 7

View of the MDI PEL Concentration at 38°C - Cross Section View

Position of the MDI TLV Around an MDI Spill

Temperature = 38°C

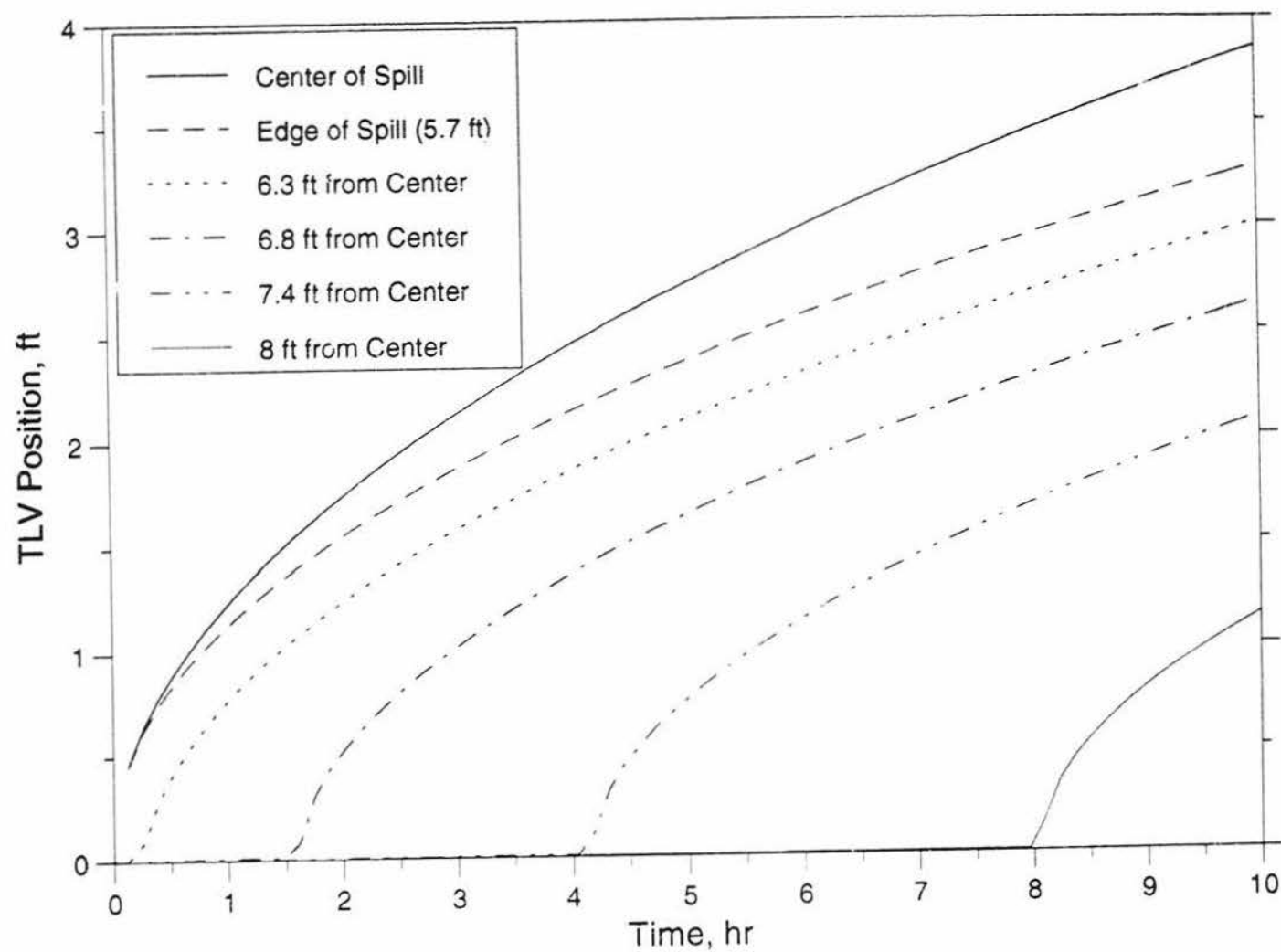


Figure 8

View of the MDI PEL Concentration at 38°C - Temporal View

Appendix A

Finite Difference Method

Peaceman-Rachford (ADI)

Using central difference approximations, the finite difference forms for the partial derivatives are:

$$\frac{\partial \chi}{\partial \tau} \sim \frac{\chi_{i,j}^{k+1} - \chi_{i,j}^k}{\Delta \tau} \quad (\text{A-1})$$

$$\frac{1}{\eta} \frac{\partial \chi}{\partial \eta} \sim \frac{1}{j \Delta \eta} \frac{\chi_{i,j+1} - \chi_{i,j-1}}{2 \Delta \eta} \quad (\text{A-2})$$

$$\frac{\partial^2 \chi}{\partial \eta^2} \sim \frac{\chi_{i,j+1} - 2\chi_{i,j} + \chi_{i,j-1}}{\Delta \eta^2} \quad (\text{A-3})$$

$$\frac{\partial^2 \chi}{\partial \xi^2} \sim \frac{\chi_{i+1,j} - 2\chi_{i,j} + \chi_{i-1,j}}{\Delta \xi^2} \quad (\text{A-4})$$

where j is the radial grid point index,
 i is the vertical grid point index,
 k is the time step index,
 $\Delta \eta$ is the grid spacing in the radial direction (constant),
 $\Delta \xi$ is the grid spacing in the vertical direction (constant) and
 $\Delta \tau$ is the time step size.

The grid spacing in the radial and vertical directions do not have to be the same.

For the ADI method, the integration is carried out in two stages. In the first stage, an intermediate solution is calculated at $1/2$ of the time step for only the vertical direction. Equation (8) in finite difference form for the intermediate time step is

$$\frac{\chi_{i,j}^* - \chi_{i,j}^k}{\frac{\Delta \tau}{2}} = \frac{\chi_{i,j+1}^k - 2\chi_{i,j}^k + \chi_{i,j-1}^k}{\Delta \eta^2} + \frac{1}{j \Delta \eta} \frac{\chi_{i,j+1}^k - \chi_{i,j-1}^k}{2 \Delta \eta} + \frac{\chi_{i+1,j}^* - 2\chi_{i,j}^* + \chi_{i-1,j}^*}{\Delta \xi^2} \quad (\text{A-5})$$

where $*$ is the intermediate time step solution and k is the starting time step values. For the second stage, the final solution at the end of the time step is calculated using only the radial direction. The finite difference equation for this step is given by

$$\frac{\chi_{i,j}^{k+1} - \chi_{i,j}^*}{\frac{\Delta \tau}{2}} = \frac{\chi_{i,j+1}^{k+1} - 2\chi_{i,j}^{k+1} + \chi_{i,j-1}^{k+1}}{\Delta \eta^2} + \frac{1}{j \Delta \eta} \frac{\chi_{i,j+1}^{k+1} - \chi_{i,j-1}^{k+1}}{2 \Delta \eta} + \frac{\chi_{i+1,j}^* - 2\chi_{i,j}^* + \chi_{i-1,j}^*}{\Delta \xi^2} \quad (\text{A-6})$$

Rearranging equations (A-5) and (A-6)

$$-\chi_{i+1,j}^* + 2\left(1 + \frac{1}{\lambda}\right)\chi_{i,j}^* - \chi_{i-1,j}^* = \frac{\rho}{\lambda} \left[\left(1 + \frac{1}{2j}\right)\chi_{i,j+1}^* + 2\left(\frac{1}{\rho} - 1\right)\chi_{i,j}^* + \left(1 - \frac{1}{2j}\right)\chi_{i,j-1}^* \right] \quad (\text{A-7})$$

$$-\left(1 + \frac{1}{2j}\right)\chi_{i,j+1}^{k+1} + 2\left(1 + \frac{1}{\rho}\right)\chi_{i,j}^{k+1} - \left(1 - \frac{1}{2j}\right)\chi_{i,j-1}^{k+1} = \frac{\lambda}{\rho} \left[\chi_{i+1,j}^* + 2\left(\frac{1}{\lambda} - 1\right)\chi_{i,j}^* + \chi_{i-1,j}^* \right] \quad (\text{A-8})$$

where $\lambda = \Delta\tau / \Delta\eta^2$ and $\rho = \Delta\tau / \Delta\xi^2$. Equations (A-7) and (A-8) are solved in sequence to give the final solution at the next time step.

At the centerline, $\eta = 0$ ($j = 0$), the finite difference form of Equation (6) is used as a boundary condition which gives the following two equations for each half time step:

$$-\chi_{i+1,0}^* + 2\left(1 + \frac{1}{\lambda}\right)\chi_{i,0}^* - \chi_{i-1,0}^* = \frac{\rho}{\lambda} \left[4\chi_{i,1}^* + 2\left(\frac{1}{\rho} - 2\right)\chi_{i,0}^* \right] \quad (\text{A-9})$$

$$-4\chi_{i,1}^{k+1} + 2\left(2 + \frac{1}{\rho}\right)\chi_{i,0}^{k+1} = \frac{\lambda}{\rho} \left[\chi_{i+1,0}^* + 2\left(\frac{1}{\lambda} - 1\right)\chi_{i,0}^* + \chi_{i-1,0}^* \right] \quad (\text{A-10})$$

The boundary condition at the surface of the spill, $\xi = 0$ ($i = 0$), has $\chi_{0,j} = 1$ for all points within the radius of the spill, $\eta \leq 1$ ($j \leq J$; $J = 1/\Delta\eta$). For the rest of the ground surface, $\xi = 0$ ($i = 0$) and $\eta > 1$ ($j > J$) the flux into the ground is zero. This results in the next equations for each half time step:

$$-2\chi_{1,j}^* + 2\left(1 + \frac{1}{\lambda}\right)\chi_{0,j}^* = \frac{\rho}{\lambda} \left[\left(1 + \frac{1}{2j}\right)\chi_{0,j+1}^* + 2\left(\frac{1}{\rho} - 1\right)\chi_{0,j}^* + \left(1 - \frac{1}{2j}\right)\chi_{0,j-1}^* \right] \quad (\text{A-11})$$

$$-\left(1 + \frac{1}{2j}\right)\chi_{0,j+1}^{k+1} + 2\left(1 + \frac{1}{\rho}\right)\chi_{0,j}^{k+1} - \left(1 - \frac{1}{2j}\right)\chi_{0,j-1}^{k+1} = \frac{\lambda}{\rho} \left[2\chi_{1,j}^* + 2\left(\frac{1}{\lambda} - 1\right)\chi_{0,j}^* \right] \quad (\text{A-12})$$

At the edge of the finite difference grid in the radial direction, $j = N$, the boundary condition which was imposed allowed no flux through that line. This acts as a solid wall. In this case the finite difference equations for each half step of the simulation are:

$$-\chi_{i+1,N}^* + 2\left(1 + \frac{1}{\lambda}\right)\chi_{i,N}^* - \chi_{i-1,N}^* = \frac{\rho}{\lambda} \left[2\chi_{i,N-1}^* + 2\left(\frac{1}{\rho} - 1\right)\chi_{i,N}^* \right] \quad (\text{A-13})$$

$$-2\chi_{i,N-1}^{k+1} + 2\left(1 + \frac{1}{\rho}\right)\chi_{i,N}^{k+1} = \frac{\lambda}{\rho} \left[\chi_{i+1,N}^* + 2\left(\frac{1}{\lambda} - 1\right)\chi_{i,N}^* + \chi_{i-1,N}^* \right] \quad (\text{A-14})$$

At the top of the grid in the vertical direction, $i = M$, there is no flux through that line. This creates a solid roof. The finite difference equations at that boundary for each half step are:

$$-2\chi_{M-1,j}^* + 2\left(1 + \frac{1}{\lambda}\right)\chi_{M,j}^* = \frac{\rho}{\lambda} \left[\left(1 + \frac{1}{2j}\right)\chi_{M,j+1}^* + 2\left(\frac{1}{\rho} - 1\right)\chi_{M,j}^* + \left(1 - \frac{1}{2j}\right)\chi_{M,j-1}^* \right] \quad (\text{A-15})$$

$$-\left(1 + \frac{1}{2j}\right)\chi_{M,j+1}^{k+1} + 2\left(1 + \frac{1}{\rho}\right)\chi_{M,j}^{k+1} - \left(1 - \frac{1}{2j}\right)\chi_{M,j-1}^{k+1} = \frac{\lambda}{\rho} \left[2\chi_{M-1,j}^* + 2\left(\frac{1}{\lambda} - 1\right)\chi_{M,j}^* \right] \quad (\text{A-16})$$

At each corner of the grid special conditions exist which must be handled separately. At the upper right corner ($j = N$; $i = M$) and the lower right corner ($j = N$; $i = 0$) the flux is zero in both the radial and vertical directions. The finite difference equations at these points are:

(Upper Right - $j = N$; $i = M$)

$$-\chi_{M-1,N}^* + \left(1 + \frac{1}{\lambda}\right)\chi_{M,N}^* = \frac{\rho}{\lambda} \left[\chi_{M,N-1}^* + \left(\frac{1}{\rho} - 1\right)\chi_{M,N}^* \right] \quad (\text{A-17})$$

$$-\chi_{M,N-1}^{k+1} + \left(1 + \frac{1}{\rho}\right)\chi_{M,N}^{k+1} = \frac{\lambda}{\rho} \left[\chi_{M-1,N}^* + \left(\frac{1}{\lambda} - 1\right)\chi_{M,N}^* \right] \quad (\text{A-18})$$

(Lower Right - $j = N$; $i = 0$)

$$-\chi_{1,N}^* + \left(1 + \frac{1}{\lambda}\right)\chi_{0,N}^* = \frac{\rho}{\lambda} \left[\chi_{0,N-1}^* + \left(\frac{1}{\rho} - 1\right)\chi_{0,N}^* \right] \quad (\text{A-19})$$

$$-\chi_{0,N-1}^{k+1} + \left(1 + \frac{1}{\rho}\right)\chi_{0,N}^{k+1} = \frac{\lambda}{\rho} \left[\chi_{1,N}^* + \left(\frac{1}{\lambda} - 1\right)\chi_{0,N}^* \right] \quad (\text{A-20})$$

At the centerline ($j = 0$) the finite difference form of equation (6) must be used along with the condition that at the upper left point ($j = 0$; $i = M$) that there is no flux in both the radial and the vertical directions. The finite difference equations for this point become:

$$-\chi_{M+1,0}^* + \left(1 + \frac{1}{\lambda}\right) \chi_{M,0}^* = \frac{\rho}{\lambda} \left[2\chi_{M,1}^* + \left(\frac{1}{\rho} - 2\right) \chi_{M,0}^* \right] \quad (\text{A-21})$$

$$-2\chi_{M,1}^{*+1} + \left(2 + \frac{1}{\rho}\right) \chi_{M,0}^{*+1} = \frac{\lambda}{\rho} \left[\chi_{M-1,0}^* + \left(\frac{1}{\lambda} - 1\right) \chi_{M,0}^* \right] \quad (\text{A-22})$$

Appendix B

Program Listing

```

/***** Program "SPILL" *****/

```

This is a two space dimension dynamic simulation model of the concentrations around a circular spill. The diffusion is assumed to be in stagnant air from a material with very low volatility (trace component).

Version 1.0 31-October-1993 G. Roginski

```

*****/

```

```

/* Include the required libraries */

```

```

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

```

```

/* Define program parameters */

```

```

#define MAXCOLS      60
#define MAXROWS      100
#define MAXWORK      100
#define MAX          100
#define MAXLIMITS 10

```

```

/* Define function prototypes */

```

```

void FindLimits      ( float x[][MAXCOLS], int, int, int, int,
                      float *l, int, float l2[][MAXCOLS]);
float f_minus        ( int );
float f_plus         ( int );
void InitArrays      ( float x[][MAXCOLS], float x1[][MAXCOLS],
                      int, int, int );
float Interpolate     ( float, float, float, float, float );
void PrintArray       ( float x[][MAXCOLS], int, int, int, int );
void PrintLimits      ( FILE *fp, float, float x[][MAXCOLS], int, int );
void SetColumnArrays  ( float *s, float *d, float *u, float *r,
                      int, int, int, int,
                      float, float, float, float x[][MAXCOLS] );
void SetRowArrays     ( float *s, float *d, float *u, float *r,
                      int, int, int, int,
                      float, float, float, float x[][MAXCOLS] );
void StoreColumn      ( float *s, float x[][MAXCOLS], int, int, int );
void StoreRow         ( float *s, float x[][MAXCOLS], int, int, int );
float TableLookUp     ( float, float x[][MAXCOLS], int, int, int );
void Tridiag          ( float *s, float *d, float *u, float *r, float *sol,
                      int, int );

```

```

/***** Main Routine *****/

```

```

main()

```

```

{

```

```

    float x[MAXROWS][MAXCOLS], x_star[MAXROWS][MAXCOLS];
    int ncols, nrows, nspill, nspill1;
    int nlimits;
    float limits[MAXLIMITS], limit_positions[MAXLIMITS][MAXCOLS];
    float d_xi, d_eta, d_tau, r, Dab, r_convert, z_convert;
    float tau, tau_end, t_convert, t_actual;
    int print_interval, n_print;
    int n, m, first, last;
    int i, j, k;
    float d_lambda_plus, d_lambda_minus, d_rho_plus, d_rho_minus,
          ratio, lambda, rho;
    float sub[MAXWORK], diag[MAXWORK], super[MAXWORK], rhs[MAXWORK],
          solution[MAXWORK];

```

```

FILE *fp_in, *fp_out;

/* Read the input data */
/*
Input variables have the following meanings...
d_xi    the increment in the "z" direction (dimensionless) (xi = z/R)
d_eta   the increment in the "r" direction (dimensionless) (eta = r/R)
d_tau   the time increment (dimensionless) (tau = Dab*t/R^2)
tau_end the ending time (dimensionless)
r       the radius of the spill (R) (feet)
Dab     the diffusivity of A in B (ft^2/min)
print_interval
        the print interval, i.e., the number of steps between printouts
ncols   the number of columns in the simulation grid
nrows   the number of rows in the simulation grid
nlimits the number of exposure limit positions to be found by interpolation
limits  the exposure limits (one for each nlimits) as a fraction of the
        vapor pressure
*/

if( (fp_in = fopen("a:spill.dat","r")) == NULL )
{
    printf("\nCannot open the input file.");
    exit(EXIT_FAILURE);
}

fscanf(fp_in,"%f %f %f %f %f %f %i %i %i", &d_xi, &d_eta, &d_tau,
        &tau_end, &r, &Dab, &print_interval, &ncols, &nrows);
fscanf(fp_in,"%i", &nlimits);
for ( i=0; i<nlimits; i++ ) fscanf(fp_in,"%f", limits+i);

fclose(fp_in);

/* Create an output file to hold the exposure limits positions */

if( (fp_out = fopen("limits.out","w")) == NULL )
{
    printf("\nCannot open the output file.");
    exit(EXIT_FAILURE);
}

/* Print the input data */

printf("\nInput Data\n\n");
printf("d_xi= %f\n",d_xi);
printf("d_eta= %f\n",d_eta);
printf("d_tau= %f\n",d_tau);
printf("tau_end= %f\n",tau_end);
printf("r = %f\n", r);
printf("Dab = %f\n", Dab);
printf("print_interval = %i\n", print_interval);
printf("ncols, nrows = %i %i\n",ncols,nrows);
printf("nlimits = %i\n",nlimits);
printf("limits = ");
for ( i=0; i<nlimits ;i++) printf("%f ",limits[i]);

nspill = (int)(1.0/d_eta + 0.5); /* Index to the last point inside of the spill
*/

n = ncols-1; /* Index to the last point in a row */
m = nrows-1; /* Index to the last point in a column */

/* The simulation grid in the eta (r) and xi (z) plane is as follows:

m + + + . . . . + +

```

```

^ m-1 + + + . . . . + +
| . . . . . . . . . .
xi . . . . . . . . . .
(z) . . . . . . . . . .
2 + + + . + + . . . + +
1 + + + + + . . . + +
0 + + + + + . . . + +
0 1 2 . nspill nspill+1 . . . n-1 n
eta (r) ->

```

```

*/
/* Calculate the simulation parameters */

lambda = d_tau/(d_xi*d_xi);
rho     = d_tau/(d_eta*d_eta);
ratio   = rho/lambda;

d_lambda_plus   = 2.0*(1.0/lambda + 1.0);
d_lambda_minus  = 2.0*(1.0/lambda - 1.0);
d_rho_plus      = 2.0*(1.0/rho + 1.0);
d_rho_minus     = 2.0*(1.0/rho - 1.0);

r_convert = d_eta * r;          /* Conversion from d_eta to "r" in feet */
z_convert = d_xi  * r;          /* Conversion from d_xi to "z" in feet */
t_convert = r*r / Dab;          /* Conversion from tau to "t" in sec */

printf("\n\nCalculated Parameters\n\n");
printf("lambda= %f\n", lambda);
printf("rho= %f\n", rho);
printf("ratio= %f\n", ratio);
printf("d_lambda_plus= %f\n", d_lambda_plus);
printf("d_lambda_minus= %f\n", d_lambda_minus);
printf("d_rho_plus= %f\n", d_rho_plus);
printf("d_rho_minus= %f\n\n", d_rho_minus);

printf("r_convert = %f * d_eta [=] feet\n", r_convert);
printf("z_convert = %f * d_xi  [=] feet\n", z_convert);
printf("t_convert = %f * tau   [=] min\n", t_convert);

tau      = 0.0;
n_print  = 0;

t_actual = 0.0;

/* Initialize the data arrays "x" is the mole fraction at a grid point
   "x_star" is an intermediate value
*/
InitArrays( x, x_star, m, n, nspill );

printf("\n\ntau= %f      t= %f\n", tau, t_actual);
PrintArray( x, 0, m, 0, n );

/* Perform the simulation
   The numerical is the Alternating Direction Implicit (ADI) method of
   Peaceman and Rachford
*/

while( tau < tau_end )
{
    /* First calculate an intermediate solution for each column */
    for ( j = 0; j<=n; j++ )
    {
        if( j<=nspill )
        {

```

```

        first = 1;
        last  = m;
    }
    else
    {
        first = 0;
        last  = m;
    }
    SetColumnArrays(sub,diag,super,rhs,j,n,nspill,
                    first,last,
                    d_lambda_plus,d_rho_minus,ratio,x);
    Tridiag(sub,diag,super,rhs,solution,first,last);
    StoreColumn(solution,x_star,j,first,last);
}

/* Next calculate the final solution based on the rows */
for ( i = 0; i<=m; i++)
{
    if ( i==0 )
    {
        first = nspill + 1;
        last  = n;
    }
    else
    {
        first = 0;
        last  = n;
    }
    SetRowArrays(sub,diag,super,rhs,i,m,nspill+1,
                 first,last,
                 d_rho_plus,d_lambda_minus,1.0/ratio,x_star);
    Tridiag(sub,diag,super,rhs,solution,first,last);
    StoreRow(solution,x,i,first,last);
}

/* Increment the time and print counters */
tau    += d_tau;
n_print += 1;

/* Print out the results */
if( n_print == print_interval )
{
    n_print = 0;
    t_actual = tau * t_convert;
    printf("tau= %f    t= %f\n",tau, t_actual);
    PrintArray( x, 0, m, 0, n );
    FindLimits( x, 0, m, 0, n, limits, nlimits, limit_positions );
    PrintLimits(fp_out,tau,limit_positions,nlimits,n);
}

fclose(fp_out);
return;
}

/***** Functions *****/
void FindLimits( float x[MAXROWS][MAXCOLS], int first_row, int last_row,
                int first_col, int last_col, float limits[MAXLIMITS],
                int nlimits, float limit_positions[MAXLIMITS][MAXCOLS])
{
    /* Routine to locate the position of the exposure limits in each column

```

of the simulation.

Version 1.0 31-October-1993 G. Roginski

Variables have the following meanings...

x an array containing the solution
 first_row the index number for the first row to be used in the search
 last_row the index number for the last row to be used in the search
 first_col the index number for the first column to be used
 last_col the index number for the last column to be used
 limits an array containing the values of the exposure limits to be
 found
 nlimits the number of "limits" to be found
 limit_positions
 an array containing the locations of each of the "limits"
 in each column of "x"

```

*/
int i, j;

for ( j=first_col; j<=last_col; j++ )
{
  for ( i=0; i<nlimits; i++ )
  {
    limit_positions[i][j] = TableLookUp(limits[i],x,j,first_row,last_row);
  }
}

return;
}

float f_minus( int j )
{
  return( 1.0 - 1.0/((float)(2*j)) );
}

float f_plus( int j )
{
  return( 1.0 + 1.0/((float)(2*j)) );
}

void InitArrays( float a[MAXROWS][MAXCOLS], float b[MAXROWS][MAXCOLS],
                int m, int n, int nl )
{
  /* Routine to initialize the main and intermediate arrays.

```

Version 1.0 31-October-1993 G. Roginski

Variables have the following meanings...

a, b the arrays to be initialized
 m the index number of the last row in "a" and "b"
 n the index number of the last column in "a" and "b"
 nl the index number of the last column in the spill region

```

*/

int i, j;

/* Initialize the arrays */

for ( i=0; i<=m; i++ )
{
  for ( j=0; j<=n; j++ )
  {
    a[i][j] = 0.0;

```

```

        b[i][j] = 0.0;
    }
}

/* Initialize the spill region */
for ( j=0; j<=n1; j++ )
{
    a[0][j] = 1.0;
    b[0][j] = 1.0;
}

return;
}

float Interpolate ( float x, float x1, float x2, float y1, float y2 )
{
    /* Function to interpolate between two values. This function returns the
    result of the interpolation.

    Version 1.0 31-October-1993 G. Roginski

    Variables have the following meanings...
    x      the value for which the "y" value is to be found by interpolation
    x1,x2  the given "x" values
    y1,y2  the given "y" values

    + (x2, y2)
    + (x1, y1)

    */
    return( y1 + (y2 - y1)/(x2 - x1) * (x - x1) );
}

void PrintArray ( float x[MAXROWS][MAXCOLS], int first_row, int last_row,
                  int first_col, int last_col )
{
    /* Routine to print an array. (8 columns per line)

    Version 1.0 31-October-1993 G. Roginski

    Variables have the following meanings...
    x      the array to be printed
    first_row  the index to the first row of "x" to be printed
    last_row   the index to the last row of "x" to be printed
    first_col  the index to the first column of "x" to be printed
    last_col   the index to the last column of "x" to be printed

    */

    int i, j, k, l1, l2, nblocks;

    nblocks = (last_col - first_col)/8 + 1;

    for ( k=1; k<=nblocks; k++)
    {
        printf("\n      ");

        l1 = (k-1)*8 + first_col;
        l2 = ((k*8+first_col-1)>=last_col) ? last_col : k*8+first_col-1;
        for( j=l1; j<=l2; j++) printf("%5d      ", j);
        for( i=first_row; i<=last_row; i++)
        {
            printf("\n%3d      ", i);
            for( j=l1; j<=l2; j++) printf("%8.5f ", x[i][j]);
        }
    }
}

```

```

    }
    printf("\n\n");
    return;
}

void PrintLimits ( FILE *fp, float t, float limit_positions[MAXLIMITS][MAXCOLS],
                  int nlimits, int n )
{
    /* Routine to write the exposure limit limit postions to a file.

        Version 1.0  31-October-1993  G. Roginski

    Variables have the following meanings...
    fp      FILE pointer to the output stream
    t        the time of the printout
    limit_positions
        an array containing the locations of the exposure limits
    nlimits  the number of limits to be printed
    n        an index to the last column in "limit_positions" to be printed

    Each line of the printout begins with the time "t" followed by the position
    of the limit in each column.
    */

    int i,j;

    for (i=0; i<nlimits; i++)
    {
        fprintf(fp,"%f ",t);
        for (j=0; j<=n; j++)
        {
            fprintf(fp,"%f ",limit_positions[i][j]);
        }
        fprintf(fp,"\n");
    }
    return;
}

void SetColumnArrays ( float sub[MAXWORK], float diag[MAXWORK],
                      float super[MAXWORK], float rhs[MAXWORK],
                      int j, int n, int nspill, int first, int last,
                      float a, float b, float ratio,
                      float x[MAXROWS][MAXCOLS] )
{
    /* Routine to calculate and store the values for the column arrays to be
    used in the first stage of the simulation.

        Version 1.0  31-October-1993  G. Roginski

    Variables have the following meanings...
    sub      an array containing the subdiagonal matrix elements (output)
    diag     an array containing the diagonal matrix elements (output)
    super    an array containing the superdiagonal matrix elements (output)
    rhs      an array containing the right hand side of the matrix (output)
    j        the index of the column to be used (input)
    nspill   the index of the last column inside of the spill (input)
    n        the index of the last column in the simulation (input)
    first    the index of the first row in the column to be included (input)
    last     the index of the last row in the column to be included (input)
    a        the value of the diagonal element in the matrix (input)
    b        the value of the diagonal element used in the calculation of
    the right hand side vector (input)
    ratio    the ratio of rho/lambda (input)
    x        an array containing the concentration data (input)

```

```

*/
int i;
for ( i=first; i<=last; i++ )
{
    diag[i] = a;
    sub[i] = -1.0;
    super[i] = -1.0;

    if( j==0 )
    {
        rhs[i] = ratio*(4.0*x[i][1] + (b - 2.0)*x[i][0]);
        if( i==first ) rhs[i] += x[0][0];
    }

    if( j>0 && j<n )
    {
        rhs[i] = ratio*(f_plus(j)*x[i][j+1] + b*x[i][j] +
                        f_minus(j)*x[i][j-1] );
        if( j<=nspill && i==first ) rhs[i] += x[0][j];
    }

    if( j==n )
    {
        rhs[i] = ratio*(2.0*x[i][n-1] + b*x[i][n]);
    }
}

if( j<=nspill ) sub[last] = -2.0;
if( j>nspill )
{
    super[first] = -2.0;
    sub[last] = -2.0;
}
return;
}

void SetRowArrays ( float sub[MAXWORK], float diag[MAXWORK],
                   float super[MAXWORK], float rhs[MAXWORK],
                   int i, int m, int nspill, int first, int last,
                   float a, float b, float ratio,
                   float x[MAXROWS][MAXCOLS] )
{
    /* Routine to calculate and store the values for the row arrays to be
       used in the second stage of the simulation.

       Version 1.0 31-October-1993 G. Roginski

       Variables have the following meanings...
       sub    an array containing the subdiagonal matrix elements (output)
       diag   an array containing the diagonal matrix elements (output)
       super  an array containing the superdiagonal matrix elements (output)
       rhs    an array containing the right hand side of the matrix (output)
       i      the index of the row to be used (input)
       m      the index of the last row in the simulation (input)
       nspill the index of the first column outside of the spill (input)
       first  the index of the first column in the row to be included (input)
       last   the index of the last column in the row to be included (input)
       a      the value of the diagonal element in the matrix (input)
       b      the value of the diagonal element used in the calculation of
              the right hand side vector (input)
       ratio  the ratio of lamda/rho (input)
       x      an array containing the concentration data (input)

    */
    int j;

```

```

for ( j=first; j<=last; j++ )
{
    diag[j] = a;
    if( j == 0 )
    {
        diag[j] += 2.0;
    }
    else
    {
        sub[j] = -f_minus(j);
        super[j] = -f_plus(j);
    }

    if( i==0 )
    {
        rhs[j] = ratio*(2.0*x[1][j] + b*x[0][j]);
        if( j==nspill1 ) rhs[j] += f_minus(j)*x[0][j-1];
    }

    if( i>=1 && i<m )
    {
        rhs[j] = ratio*(x[i+1][j] + b*x[i][j] + x[i-1][j] );
    }

    if( i==m )
    {
        rhs[j] = ratio*(2.0*x[m-1][j] + b*x[m][j]);
    }
}

if( i==0 ) sub[last] = -2.0;
if( i>=1 )
{
    super[first] = -4.0;
    sub[last] = -2.0;
}
return;
}

void StoreColumn( float solution[MAXWORK], float x_star[MAXROWS][MAXCOLS],
                 int j, int first, int last )
{
    /* Routine to store the results of the column decomposition in the
       corresponding column of the results array.

       Version 1.0 31-October-1993 G. Roginski

       Variables have the following meanings...
       solution      an array containing the solution of the equations
                     for a single column in the grid (input)
       x_star        an array containing the intermediate values for each
                     column (input/output)
       j             an index to the column to be stored (input)
       first         an index to the first location to be stored (input)
       last         an index to the last location to be stored (input)
    */
    int i;

    for ( i=first; i<=last; i++ )
    {
        x_star[i][j] = solution[i];
    }
    return;
}

```

```
void StoreRow ( float solution[MAXWORK], float x[MAXROWS][MAXCOLS],
               int i, int first, int last )
```

```
{
/* Routine to store the results of the column decomposition in the
   corresponding column of the results array.
```

```
Version 1.0 31-October-1993
```

```
Variables have the following meanings...
```

```
  solution    an array containing the solution of the equations
               for a single row in the grid (input)
  x            an array containing the final values for each
               row (input/output)
  i            an index to the row to be stored (input)
  first       an index to the first location to be stored (input)
  last        an index to the last location to be stored (input)
```

```
*/
```

```
int j;
```

```
for ( j=first; j<=last; j++ )
```

```
{
  x[i][j] = solution[j];
}
```

```
return;
```

```
}
```

```
float TableLookUp( float value, float x[MAXROWS][MAXCOLS], int column,
                  int first_row, int last_row )
```

```
{
```

```
/* Routine to interpolate in a column of "x" to find the location of "value".
   The values in the columns of "x" are assumed to be in decending order.
   If the value is greater than the first row value to be checked then the
   first row value is returned. If the value is less than the last row value
   to be checked than the returned value is an extrapolation.
```

```
Version 1.0 31-October-1993 G. Roginski
```

```
Variables have the following meanings...
```

```
  value       the value to be located in "x"
  x           an array where value is to be found
  column      the column of "x" to be used in the search
  first_row   the first row of "x" to be used in the search
  last_row    the last row of "x" to be used in the search
```

```
*/
```

```
int i;
```

```
if(value>x[first_row][column]) return( x[first_row][column] );
for (i=first_row+1; i<=last_row; i++)
```

```
{
  if(value>x[i][column]) return( Interpolate( value, x[i-1][column],
                                              x[i][column],
                                              (float)(i-1), (float)i ) );
}
```

```
return( Interpolate( value, x[last_row-1][column], x[last_row][column],
                    (float)(last_row-1), (float)last_row ) );
```

```
}
```

```
void Tridiag( float sub[MAX], float diag[MAX], float super[MAX],
             float rhs[MAX], float solution[MAX], int first, int last)
```

```
/* This is a routine for solving a system of linear simultaneous equations
   having a tridiagonal coefficient matrix.
```

```
Version 1.0 31-October-1993 G. Roginski
```

sub	an array containing the subdiagonal coefficients (input)
diag	an array containing the diagonal coefficients (input)
super	an array containing the superdiagonal coefficients (input)
rhs	an array containing the right hand side vector (input)
solution	an array containing the solution (output)
first	the index of the first equation (input)
last	the index of the last equation (input)

```
diag[first]      super[first]      = rhs[first]
sub[first+1]    diag[first+1]    super[first+1] = rhs[first+1]
```

$$\begin{array}{ccccccc} \text{sub}[\text{last}-1] & & \text{diag}[\text{last}-1] & & \text{super}[\text{last}-1] & = & \text{rhs}[\text{last}-1] \\ & & \text{sub}[\text{last}] & & \text{diag}[\text{last}] & = & \text{rhs}[\text{last}] \end{array}$$

1

```
/* Compute the intermediate working arrays "beta" and "gamma" */
```

```

beta[first] = diag[first];
gamma[first] = rhs[first]/beta[first];

for (i=first+1 ; i<=last ; i++)
{
    beta[i] = diag[i] - sub[i]*super[i-1]/beta[i-1];
    gamma[i] = (rhs[i] - sub[i]*gamma[i-1])/beta[i];
}

```

```
/* Backsubstitute for the solution */
```

```

solution[last] = gamma[last];
for (i=last-1 ; i>=first ; i--)
{
    solution[i] = gamma[i] - super[i]*solution[i+1]/beta[i];
}
return;

```

)

Appendix C

Estimate of the MDI-Air Diffusion Coefficient

Estimate of the Diffusivity of MDI in Air

The equation of Fuller, Schettler and Giddings is used.

$$D_{AB} := \frac{0.00143 \cdot T^{1.75}}{P \cdot M_{AB}^{\frac{1}{2}} \cdot \left(v_A^{\frac{1}{3}} + v_B^{\frac{1}{3}} \right)^2}$$

Where D_{AB} is the diffusion coefficient of A in B [cm²/sec]

T is the absolute temperature [K]

M_{AB} is the weighted molecular weight between A and B

v_A, v_B are the volumes of the molecules

P is the pressure [bar]

To calculate the molecular volume of MDI requires the volume increments for the parts of the molecule. The diffusion volume increments are:

$$v_C := 15.9 \quad \text{Carbon}$$

$$v_H := 2.31 \quad \text{Hydrogen}$$

$$v_O := 6.11 \quad \text{Oxygen}$$

$$v_N := 4.54 \quad \text{Nitrogen}$$

$$v_{\text{ring}} := -18.3 \quad \text{Ring correction}$$

MDI consists of 15 carbons, 2 oxygens, 10 hydrogens, 2 nitrogens and 2 aromatic rings.

$$v_{\text{MDI}} := 15 \cdot v_C + 2 \cdot v_O + 10 \cdot v_H + 2 \cdot v_N + 2 \cdot v_{\text{ring}}$$

For MDI:

$$v_{\text{MDI}} = 246.3$$

$$\text{MW}_{\text{MDI}} := 250.25$$

For Air (from Reid & Sherwood):

$$v_{\text{Air}} := 19.7$$

$$\text{MW}_{\text{Air}} := 28.966$$

The combined molecular weight is calculated as:

$$M_{\text{MDI_Air}} := 2 \cdot \left[\left(\frac{1}{\text{MW}_{\text{MDI}}} \right) + \left(\frac{1}{\text{MW}_{\text{Air}}} \right) \right]^{-1}$$

The desired conditions are:

Pressure=1 bar

$P := 1$

Temperature=20°C

$T := 20 + 273.15 \text{ K}$

$$D_{\text{MDI_Air}} := \frac{0.00143 \cdot T^{1.75}}{P \cdot M_{\text{MDI_Air}}^{\frac{1}{2}} \cdot \left(v_{\text{MDI}}^{\frac{1}{3}} + v_{\text{Air}}^{\frac{1}{3}} \right)^2}$$

$$D_{\text{MDI_Air}} = 0.051 \text{ cm}^2/\text{sec}$$



CERTIFICATE OF AUTHENTICITY

THIS IS TO CERTIFY that the microimages appearing on this microfiche are accurate and complete reproductions of the records of U.S. Environmental Protection Agency documents as delivered in the regular course of business for microfilming.

Data produced

9

22

97

Marcia Tubolino

(Month)

(Day)

(Year)

Camera Operator

Place

Syracuse

New York

(City)

(State)